



**STOP
PRESS**

DRAGON

EDITORIAL

Welcome to the third edition of Stop Press which, like its predecessors, has been sent to all those Dragon owners who returned their guarantee card. Those who have not (and there are many) are not as yet registered members of Dragon Users Club and will not be on the mailing list. If you know of anybody in this category, gently prod them to return their card so that future issues of Stop Press may be directed to them.

Though not (as yet) a fully-fledged magazine, Stop Press continues, in this and future issues, to offer Dragon owners pages of program-oriented material. Programs which (a) do something interesting (b) illustrate programming techniques and (c) can be typed into Dragon in a short time, are ideal for inclusion in Stop Press. Most of the material presented falls into at least two of these categories!

Two features in the last issue have their counterparts within. Indeed we hope that Machine Code Corner and the Young User Pages will be regular features, in the belief that there are many Dragon users who will appreciate a gentle introduction to machine code. MCC (nothing to do with cricket!) continues the series by showing how the commands introduced last

time can be harnessed to perform high-speed, high-resolution graphics. As each issue is published, readers will be able to accumulate sufficient expertise to make modest use of machine code in their programs.

This does not deny the fact that Dragon's BASIC interpreter offers a rich vocabulary of commands for constructing programs, whether they be games or whatever. Two of the commands, SOUND and PLAY, are explored in the YOUNG USER PAGES which also includes a puzzle and a competition. Mr W Harrold sent a contribution (Patterns) which exploits LINE. He and others will be interested in 'Take 1000 lines' where other intriguing aspects of the LINE command are investigated. Incidentally the pictures in this and the previous issue were not produced by ruler and pencil but by Dragon and a Hewlett Packard graph plotter.

Another contribution, this time from a young user in Cardiff, is a game of skill called 'Chicane'. Full marks to Gareth Rowlands for a superbly designed game and a compact program. We look forward to many other such contributions from young and old alike.

The summer months may see a lull in programmers' activities as they take advantage of the hot

weather! But don't forget to send your entries for the 'Draw a Dragon Logo' competition (see Issue 2) by the end of July please, to the editorial address.

Mike Gatty-Hyde,
Dragon Data Ltd, Keween Industrial Estate,
Mangots, Post Tatten, West Gloucestershire

As for Dragon Data, every month, hot weather or otherwise, is a month of intense activity in preparing for the launching of new ventures and the consolidation of software for Dragon 32. Stop Press will play an increasing role in providing up to date information in the months ahead.

PRICE CHANGE

On the 23 May the retail price of Dragon 32 was reduced to £175 including V.A.T. At the same time the following reductions were made: Joysticks selection 2, Graphic Animator, examples from the manual, Dragon Mountain and Dragon Selection 1 down from £7.95 to £4.95; Ghost Attack cartridge down from £24.95 to £19.95. These price adjustments are to be enhanced by the release of approximately 25 new titles over the next few weeks with many more to follow shortly thereafter. Also note that the Dragon disc drive and operating system was launched at the recent Computer Fair at Earls Court, and disc-based software will be released shortly.

MACHINE CODE CORNER



An area of programming in which the effect of machine code is particularly striking is the animation of graphics. To see the limitations of BASIC in this respect, try keying in this short program:

```
10 PMODE84,1,PC1,SCREEN1,B,COLOR8,I
20 CIRCLE16,28,3
30 DIMA16
40 GET16,16,-16,28,4,0
50 POKE16,110238-POKE16,0+38,28,1,POKE16,NEXT
60 GOTOM16
```

A small circle is drawn on the left of the screen, and moved, by the finest possible increments, to the right. The "animation" takes about 15 seconds. Our more athletic readers may like to convert the circle into a small (by the way, note how small the array dimension is - it needs to be about a fortieth of the area of the "GET" in pixels, which is 21x21 = 441.)

Well, how much can this be improved by using machine code? Try this one:

```
10 DATA 18,8,7,48,8,38,44,08,16,38,38,1,
21,38,3
20 DATA 38,38,58,28,72,44,28,38,31,21,18,38,38,38,31,28
30 POKE16,110238-POKE16,0+38,VAL16,17+38
40 NEXT
50 PMODE84,1,PC1,SCREEN1,B,COLOR8,I
60 CIRCLE16,28,3
70 DEC38,38
80 GOTOM16
```

This, of course, is really two short programs. Lines 10-38 POKE in the machine code, and lines 40-70 execute it. Even with the BREAD-POKE loop the program is finished in one second. The action itself takes one fifth of a second. After the first RUN, the machine code will be in position, so RUN40 will give you an action replay.

This is obviously faster than you will need for most purposes, and can be slowed by the use of delay loops. As with the BASIC program, every intermediate position is occupied, so a completely smooth movement is possible at any speed. The BASIC can be speeded up by leaving out some of the intermediate positions, but this results in a jerky movement.

Before we look in detail at animations of this sort, we must investigate the means by which Dragon stores graphic information. We shall concentrate on PMODE84,1. This makes use of memories \$6000 to \$1DFF or 16384 to 32768, i.e. a total of 6144 bytes. In this mode, each of the 256x192 pixels is either "on" (green or buff) or "off" (black). If it is "on" it has value 1, if it is "off" it has value 0. Please note

grouped together in sets of eight, so the top line of the screen is made up of 32 sets of 8 pixels, and each set is represented by one byte of memory. For example, if the first eight pixels are "on", \$00, off, off, on, on, off, off, then \$6000 will contain the binary number 11001000 (decimal 284). This can be seen by running

```
10 PMODE84,SCREEN1,B,PC1,65,988,284
20 GOTOM28
```

[PC1,65 turns the whole graphics-screen on i.e. it places binary 11111111, or decimal 255, in each memory, \$6000 to \$1DFF.] So the rows of the screen are contained in memories \$6000-\$61FF, \$6200-\$63FF, \$6400-\$65FF etc.

From this discussion, you may have concluded that it is easier to move things up and down than sideways - and you would be right. The byte configuration remains the same in vertical movement. So let's see if we can make the balloon go up, in a controlled sort of way.

Let's start in BASIC:

```
10 PMODE84,1,PC1,SCREEN1,B,COLOR8,I
20 CIRCLE16,18,1,PAINT16,18,18
30 LINE16,18,18-(18,18),PAINT
40 LINE16,18-(18,18),PAINT,18
50 DIMA16,18-(18,18)-(18,18),18,18,A/S
60 POKE16,110238-POKE16,0+38
70 -18,18,1,PAINT,NEXT
80 GOTOM16
```

Our balloon drawing is (conveniently) confined to the 18th and 17th bytes in rows 163 to 171. As with GET and PUT it is useful to include the bottom blank row to avoid the problem of "rubbing out" the bottom row of the picture as it moves up. Row 163 starts at byte \$6000 + 32*163 = \$1A62, so the top of the picture is \$1A6F and \$1A70, then \$1A7F and \$1A80 etc.

The following assembly language routine will do the job:

			Machine Code
1	LDY	#1A6F	18 98 1A 6F
2	LOP1	18,88	3 38 44
3	LOP2	18,0	3C 84
4	STD	-32,X	40 98 38
5	LEA	32,X	38 98 28
6	CMPO	#1A7F	18 93 FF FF
7	RSR	LOP3	28 F2
8	UDAY	-32,Y	3C 98 68
9	CMPT	#1A6F	18 92 88 8F
10	RSR	LOP1	28 E7
11	RTS		28

One command you may not have met before is Load Effective Address (LEA). The statement `LEAX, Y` may be compared with `LDT, Y` but instead of loading into X the data indexed by Y, the actual address (the "value" of Y) is loaded into X. The opposites for LEAX and LEAT are, respectively, `LDAX, Y` and `LDAT, Y`.

The numbers in the operand field, prior to the comma, in lines 4, 5 and 6 are called offsets. They cause the command to be applied to the register value + offset. In other words, line 4 stores the value of D in the memory whose address is 32 less than the value in X. line 5 increments the value in X by 32. line 6 decrements Y by 32.

The byte following an opcode which requires a register as part of its operand is called a postbyte. We met one in the last edition: JX+16 is represented by postbyte $\$80$. For our present program, we need to know that X and Y are respectively $\$90$ and $\$44$ (with no offset) and with offsets in the range -16 to 127 or -128 to -1 , the postbytes are $\$80$ and $\$40$ (for X and Y), followed by another byte - the offset. Negative offsets are $\$FF$ to -1 , $\$FF$ to -1 etc.

Our BASH program, incorporating the reaction code, is:

Finally, to control the speed, you insert between lines 7 and 8 the following code lines:

	Machine Code		
LDX	00FF	8E 7F FE	
LOOPS	00A0	-1.0	3E 1F
	00E1	LOOPS	2E 1C

The positivity of LEA8 = 1.8 requires explanation. When the offset lies between -16 and 16, the positivity represents the offset. For offsets of 0 to 16, positivity = offset. For offsets of -16 to -1, positivity = offset + 320.

The only other modification to the machine code is to the relative address of line 18. 26 E7 becomes 26 E8. The following program gives a controlled movement:

19 DATA 1800,74,47,39,AA,DC,04,23,00,00,
20,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
21 DATA 2307,00,77,01,00,17,36,PC,31,AA,
18,MAC,04,23,00,00
22 P001 ->T0001-BEACH1-P00200001-1,0001-1
H=0.0001 MEET
23 P000001,PC01,SCREEN1,0,COLOR1

190 CIRCLE(101,128,7,PAINT)120,100
191 LINE(101,128)---120,100,PSLT;
192 LINE(101,128)---120,100,PSLT
193 LINE(101,128)---102,100,PSLT;B
194 PSET(101,128,PASET(101,128,100,100))
195 EXEC(100)

Line 60 puts the value $5-200+200 = 1535$ into memory \$16F0/14FF. Smaller values give a faster movement; larger ones give a slower movement. The result - a very fine control over the speed of the cursor.

PCOPY with Care

When we write BASIC programs on DRAGON 32 we are well protected by a sophisticated error trap which prevents us from giving commands that cannot be carried out. As an example of this, turn on your Dragon and type PMODE4.5. The answer is PC ERROR - you can't address page 5 in PMODE4 until you have PCLEARed 8 pages (the default is 4). So type PCLEAR8 and then PMODE4.5. This is accepted.

Now type **POCLEAR4**. Back comes the answer: **PC ERROR**. You can't **POCLEAR** down to 4 pages while you are addressing page 5. So type **POCLEAR4**, and then **POC 5**.

Most commands are protected in this way. The obvious exception is the PORE command - if you PORE addressless carelessly you are likely to lose control - try typing PORE1362 (136 and 137 control the cursor position).

A less obvious problem is the PCOPY command. You might expect this to be protected in the same way as PMODE and PCLEAR - but it isn't. If you type a program in, and then (without PCLEARING) type PCOPY 1 TO 6, your program will disappear. This is because graphics page 1 has been copied to the location which held your program. Admittedly this would be a silly thing to do but the problem is more likely to arise with programs containing statements like PCOPY 1 TO 3, when it is quite easy to get the value of J wrong, so accidentally copy 1 to 3.

PATTERNS

Thanks to Mr William Harrold of Cambridge for this short program. He suggests using densities of between 3 and 6. It is a program that we have seen around in various forms. - the interference patterns caused by the discrete nature of the screen pixels are generally referred to as MOIRE patterns. They will occur when lines are plotted close together.

PARTIAL RESTORE



When a program runs on a large number of DATA statements which may have to be accessed in any order, the most convenient method to adopt is to read the whole dataset into an array and use an indexing variable to access it. This takes up a fair amount of room in the machine, however, and an alternative is to read through the whole dataset every time access is required, using the RESTORE command to return to the beginning of the list. This is undoubtedly the most efficient method of conserving memory, but is a rather cumbersome procedure. A partial RESTORE can be achieved using the DATA memory pointer, in addresses 633 and 634 (decimal 1622). By PEEKing these addresses at suitable points the first time the data are read, we can store relevant values so that the pointer can be reset to those points by POKEing at a later time. The principle is illustrated by this small program.

```
10 REM - 100000 - PEEK01/10 - PEEK02
20 READNEXT
30 CLS:PRINT#0,"WHICH STATEMENT?"
40 READ#0,VAL0H->VAL0H
50 IF VAL0H>VAL1H->VAL1H
60 >THEM
70 X=VAL0H:POKE1,000:POKE2,X0
80 READ#0,PRINT#1,0
90 FOR I=1TO4:NEXT#0
100 DATA "FIRST STATEMENT"
110 DATA "SECOND STATEMENT"
120 DATA "THIRD STATEMENT"
```

JOYSTICK GAME



This game is for two players (one on each joystick). The left joystick controls the "L" and the right joystick controls the "R". The idea is to manoeuvre your letter about the screen "running down" as many of the numbers as possible, before your opponent gets to them. You score the value of the number and your score is shown at the top of the screen. If you disappear off one edge of the screen, you will reappear immediately at the opposite edge. The numbers are 1-9 (so if you see an "8", it is just an eight and a seven together). New numbers appear from time to time to keep you busy. The time limit is controlled by the 409 in line 80.

```
10 REM JOYSTICK GAME
20 REM A.B. BAYER 1980
30 CLS:GOSUB30
40 LET I=12:GOSUB20:J7=J-100:
50 POKE1,I,I
60 LET I=18:GOSUB20:J8=J-100:
70 POKE1,I,I
80 PRINT"LEFT = I",PRINT#0,I,"RIGHT = J"
90 PRINT#1,"TIME = T"
100 SIS=0:STI=0:ST=1
```



Programs

```
100 POKE=17088
101 IF INT((100*INT=1) OR((I=57)=57))
102 THEN GOSUB10
103 JT=88022-1:GOSUB20
104 JT=1-JT:GOSUB20
105 PRINT#0,ST1:PRINT#1,ST2:PRINT#0,ST1
106 NEXTT
107 IF ST1=88168PRINT#0,ST1,"RIGHT IS THE
108 WINNER":GOTO108
109 IF ST1=88168PRINT#0,ST1,"LEFT IS THE
110 WINNER":GOTO108
111 PRINT#0,ST1,"THE RESULT IS A DRAW"
112 PRINT#0,ST1,"DO YOU WANT ANOTHER GAME?"
113 ST=1:INKEY$=0:PRINT#1,ST
114 STI=1-THEEND
115 POK1=1710000000000000
116 POK1=1710000000000000
117 POKEM000:J1=17:NEXTT:S=0-THEEND
118 J=88017POKE1,J1:17:NEXTT:S=0-THEEND
119 H=88-107POKE2/J1:H1=17:HEND=S=1-T
120 POK1=88THEEND=1
121 V=88-107POKE2/J7:V1=
122 POK1=88THEEND=2
123 POK1=88THEEND=3
124 X=88-107POKE2/J1:X1=17:XEND=S=3-T
125 POK1=88THEEND=4
126 POK1=88-107POKE2/J1:POKE1,X1,X1=17-X
127 POK1=88THEEND=5
128 POK1=88THEEND=6
129 POK1=88THEEND=7
130 POK1=88THEEND=8
131 POK1=88THEEND=9
132 POK1=88THEEND=10
133 POK1=88THEEND=11
134 POK1=88THEEND=12
135 POK1=88THEEND=13
136 POK1=88THEEND=14
137 POK1=88THEEND=15
138 POK1=88THEEND=16
139 POK1=88THEEND=17
140 POK1=88THEEND=18
141 POK1=88THEEND=19
142 POK1=88THEEND=20
143 POK1=88THEEND=21
144 POK1=88THEEND=22
145 POK1=88THEEND=23
146 POK1=88THEEND=24
147 POK1=88THEEND=25
148 POK1=88THEEND=26
149 POK1=88THEEND=27
150 POK1=88THEEND=28
151 POK1=88THEEND=29
152 POK1=88THEEND=30
153 POK1=88THEEND=31
154 POK1=88THEEND=32
155 POK1=88THEEND=33
156 POK1=88THEEND=34
157 POK1=88THEEND=35
158 POK1=88THEEND=36
159 POK1=88THEEND=37
160 POK1=88THEEND=38
161 POK1=88THEEND=39
162 POK1=88THEEND=40
163 POK1=88THEEND=41
164 POK1=88THEEND=42
165 POK1=88THEEND=43
166 POK1=88THEEND=44
167 POK1=88THEEND=45
168 POK1=88THEEND=46
169 POK1=88THEEND=47
170 POK1=88THEEND=48
171 POK1=88THEEND=49
172 POK1=88THEEND=50
173 POK1=88THEEND=51
174 POK1=88THEEND=52
175 POK1=88THEEND=53
176 POK1=88THEEND=54
177 POK1=88THEEND=55
178 POK1=88THEEND=56
179 POK1=88THEEND=57
180 POK1=88THEEND=58
181 POK1=88THEEND=59
182 POK1=88THEEND=60
183 POK1=88THEEND=61
184 POK1=88THEEND=62
185 POK1=88THEEND=63
186 POK1=88THEEND=64
187 POK1=88THEEND=65
188 POK1=88THEEND=66
189 POK1=88THEEND=67
190 POK1=88THEEND=68
191 POK1=88THEEND=69
192 POK1=88THEEND=70
193 POK1=88THEEND=71
194 POK1=88THEEND=72
195 POK1=88THEEND=73
196 POK1=88THEEND=74
197 POK1=88THEEND=75
198 POK1=88THEEND=76
199 POK1=88THEEND=77
200 POK1=88THEEND=78
201 POK1=88THEEND=79
202 POK1=88THEEND=80
203 POK1=88THEEND=81
204 POK1=88THEEND=82
205 POK1=88THEEND=83
206 POK1=88THEEND=84
207 POK1=88THEEND=85
208 POK1=88THEEND=86
209 POK1=88THEEND=87
210 POK1=88THEEND=88
211 POK1=88THEEND=89
212 POK1=88THEEND=90
213 POK1=88THEEND=91
214 POK1=88THEEND=92
215 POK1=88THEEND=93
216 POK1=88THEEND=94
217 POK1=88THEEND=95
218 POK1=88THEEND=96
219 POK1=88THEEND=97
220 POK1=88THEEND=98
221 POK1=88THEEND=99
222 POK1=88THEEND=100
223 POK1=88THEEND=101
224 POK1=88THEEND=102
225 POK1=88THEEND=103
226 POK1=88THEEND=104
227 POK1=88THEEND=105
228 POK1=88THEEND=106
229 POK1=88THEEND=107
230 POK1=88THEEND=108
231 POK1=88THEEND=109
232 POK1=88THEEND=110
233 POK1=88THEEND=111
234 POK1=88THEEND=112
235 POK1=88THEEND=113
236 POK1=88THEEND=114
237 POK1=88THEEND=115
238 POK1=88THEEND=116
239 POK1=88THEEND=117
240 POK1=88THEEND=118
241 POK1=88THEEND=119
242 POK1=88THEEND=120
243 POK1=88THEEND=121
244 POK1=88THEEND=122
245 POK1=88THEEND=123
246 POK1=88THEEND=124
247 POK1=88THEEND=125
248 POK1=88THEEND=126
249 POK1=88THEEND=127
250 POK1=88THEEND=128
251 POK1=88THEEND=129
252 POK1=88THEEND=130
253 POK1=88THEEND=131
254 POK1=88THEEND=132
255 POK1=88THEEND=133
256 POK1=88THEEND=134
257 POK1=88THEEND=135
258 POK1=88THEEND=136
259 POK1=88THEEND=137
260 POK1=88THEEND=138
261 POK1=88THEEND=139
262 POK1=88THEEND=140
263 POK1=88THEEND=141
264 POK1=88THEEND=142
265 POK1=88THEEND=143
266 POK1=88THEEND=144
267 POK1=88THEEND=145
268 POK1=88THEEND=146
269 POK1=88THEEND=147
270 POK1=88THEEND=148
271 POK1=88THEEND=149
272 POK1=88THEEND=150
273 POK1=88THEEND=151
274 POK1=88THEEND=152
275 POK1=88THEEND=153
276 POK1=88THEEND=154
277 POK1=88THEEND=155
278 POK1=88THEEND=156
279 POK1=88THEEND=157
280 POK1=88THEEND=158
281 POK1=88THEEND=159
282 POK1=88THEEND=160
283 POK1=88THEEND=161
284 POK1=88THEEND=162
285 POK1=88THEEND=163
286 POK1=88THEEND=164
287 POK1=88THEEND=165
288 POK1=88THEEND=166
289 POK1=88THEEND=167
290 POK1=88THEEND=168
291 POK1=88THEEND=169
292 POK1=88THEEND=170
293 POK1=88THEEND=171
294 POK1=88THEEND=172
295 POK1=88THEEND=173
296 POK1=88THEEND=174
297 POK1=88THEEND=175
298 POK1=88THEEND=176
299 POK1=88THEEND=177
300 POK1=88THEEND=178
301 POK1=88THEEND=179
302 POK1=88THEEND=180
303 POK1=88THEEND=181
304 POK1=88THEEND=182
305 POK1=88THEEND=183
306 POK1=88THEEND=184
307 POK1=88THEEND=185
308 POK1=88THEEND=186
309 POK1=88THEEND=187
310 POK1=88THEEND=188
311 POK1=88THEEND=189
312 POK1=88THEEND=190
313 POK1=88THEEND=191
314 POK1=88THEEND=192
315 POK1=88THEEND=193
316 POK1=88THEEND=194
317 POK1=88THEEND=195
318 POK1=88THEEND=196
319 POK1=88THEEND=197
320 POK1=88THEEND=198
321 POK1=88THEEND=199
322 POK1=88THEEND=200
323 POK1=88THEEND=201
324 POK1=88THEEND=202
325 POK1=88THEEND=203
326 POK1=88THEEND=204
327 POK1=88THEEND=205
328 POK1=88THEEND=206
329 POK1=88THEEND=207
330 POK1=88THEEND=208
331 POK1=88THEEND=209
332 POK1=88THEEND=210
333 POK1=88THEEND=211
334 POK1=88THEEND=212
335 POK1=88THEEND=213
336 POK1=88THEEND=214
337 POK1=88THEEND=215
338 POK1=88THEEND=216
339 POK1=88THEEND=217
340 POK1=88THEEND=218
341 POK1=88THEEND=219
342 POK1=88THEEND=220
343 POK1=88THEEND=221
344 POK1=88THEEND=222
345 POK1=88THEEND=223
346 POK1=88THEEND=224
347 POK1=88THEEND=225
348 POK1=88THEEND=226
349 POK1=88THEEND=227
350 POK1=88THEEND=228
351 POK1=88THEEND=229
352 POK1=88THEEND=230
353 POK1=88THEEND=231
354 POK1=88THEEND=232
355 POK1=88THEEND=233
356 POK1=88THEEND=234
357 POK1=88THEEND=235
358 POK1=88THEEND=236
359 POK1=88THEEND=237
360 POK1=88THEEND=238
361 POK1=88THEEND=239
362 POK1=88THEEND=240
363 POK1=88THEEND=241
364 POK1=88THEEND=242
365 POK1=88THEEND=243
366 POK1=88THEEND=244
367 POK1=88THEEND=245
368 POK1=88THEEND=246
369 POK1=88THEEND=247
370 POK1=88THEEND=248
371 POK1=88THEEND=249
372 POK1=88THEEND=250
373 POK1=88THEEND=251
374 POK1=88THEEND=252
375 POK1=88THEEND=253
376 POK1=88THEEND=254
377 POK1=88THEEND=255
378 POK1=88THEEND=256
379 POK1=88THEEND=257
380 POK1=88THEEND=258
381 POK1=88THEEND=259
382 POK1=88THEEND=260
383 POK1=88THEEND=261
384 POK1=88THEEND=262
385 POK1=88THEEND=263
386 POK1=88THEEND=264
387 POK1=88THEEND=265
388 POK1=88THEEND=266
389 POK1=88THEEND=267
390 POK1=88THEEND=268
391 POK1=88THEEND=269
392 POK1=88THEEND=270
393 POK1=88THEEND=271
394 POK1=88THEEND=272
395 POK1=88THEEND=273
396 POK1=88THEEND=274
397 POK1=88THEEND=275
398 POK1=88THEEND=276
399 POK1=88THEEND=277
400 POK1=88THEEND=278
401 POK1=88THEEND=279
402 POK1=88THEEND=280
403 POK1=88THEEND=281
404 POK1=88THEEND=282
405 POK1=88THEEND=283
406 POK1=88THEEND=284
407 POK1=88THEEND=285
408 POK1=88THEEND=286
409 POK1=88THEEND=287
410 POK1=88THEEND=288
411 POK1=88THEEND=289
412 POK1=88THEEND=290
413 POK1=88THEEND=291
414 POK1=88THEEND=292
415 POK1=88THEEND=293
416 POK1=88THEEND=294
417 POK1=88THEEND=295
418 POK1=88THEEND=296
419 POK1=88THEEND=297
420 POK1=88THEEND=298
421 POK1=88THEEND=299
422 POK1=88THEEND=300
423 POK1=88THEEND=301
424 POK1=88THEEND=302
425 POK1=88THEEND=303
426 POK1=88THEEND=304
427 POK1=88THEEND=305
428 POK1=88THEEND=306
429 POK1=88THEEND=307
430 POK1=88THEEND=308
431 POK1=88THEEND=309
432 POK1=88THEEND=310
433 POK1=88THEEND=311
434 POK1=88THEEND=312
435 POK1=88THEEND=313
436 POK1=88THEEND=314
437 POK1=88THEEND=315
438 POK1=88THEEND=316
439 POK1=88THEEND=317
440 POK1=88THEEND=318
441 POK1=88THEEND=319
442 POK1=88THEEND=320
443 POK1=88THEEND=321
444 POK1=88THEEND=322
445 POK1=88THEEND=323
446 POK1=88THEEND=324
447 POK1=88THEEND=325
448 POK1=88THEEND=326
449 POK1=88THEEND=327
450 POK1=88THEEND=328
451 POK1=88THEEND=329
452 POK1=88THEEND=330
453 POK1=88THEEND=331
454 POK1=88THEEND=332
455 POK1=88THEEND=333
456 POK1=88THEEND=334
457 POK1=88THEEND=335
458 POK1=88THEEND=336
459 POK1=88THEEND=337
460 POK1=88THEEND=338
461 POK1=88THEEND=339
462 POK1=88THEEND=340
463 POK1=88THEEND=341
464 POK1=88THEEND=342
465 POK1=88THEEND=343
466 POK1=88THEEND=344
467 POK1=88THEEND=345
468 POK1=88THEEND=346
469 POK1=88THEEND=347
470 POK1=88THEEND=348
471 POK1=88THEEND=349
472 POK1=88THEEND=350
473 POK1=88THEEND=351
474 POK1=88THEEND=352
475 POK1=88THEEND=353
476 POK1=88THEEND=354
477 POK1=88THEEND=355
478 POK1=88THEEND=356
479 POK1=88THEEND=357
480 POK1=88THEEND=358
481 POK1=88THEEND=359
482 POK1=88THEEND=360
483 POK1=88THEEND=361
484 POK1=88THEEND=362
485 POK1=88THEEND=363
486 POK1=88THEEND=364
487 POK1=88THEEND=365
488 POK1=88THEEND=366
489 POK1=88THEEND=367
490 POK1=88THEEND=368
491 POK1=88THEEND=369
492 POK1=88THEEND=370
493 POK1=88THEEND=371
494 POK1=88THEEND=372
495 POK1=88THEEND=373
496 POK1=88THEEND=374
497 POK1=88THEEND=375
498 POK1=88THEEND=376
499 POK1=88THEEND=377
500 POK1=88THEEND=378
501 POK1=88THEEND=379
502 POK1=88THEEND=380
503 POK1=88THEEND=381
504 POK1=88THEEND=382
505 POK1=88THEEND=383
506 POK1=88THEEND=384
507 POK1=88THEEND=385
508 POK1=88THEEND=386
509 POK1=88THEEND=387
510 POK1=88THEEND=388
511 POK1=88THEEND=389
512 POK1=88THEEND=390
513 POK1=88THEEND=391
514 POK1=88THEEND=392
515 POK1=88THEEND=393
516 POK1=88THEEND=394
517 POK1=88THEEND=395
518 POK1=88THEEND=396
519 POK1=88THEEND=397
520 POK1=88THEEND=398
521 POK1=88THEEND=399
522 POK1=88THEEND=400
523 POK1=88THEEND=401
524 POK1=88THEEND=402
525 POK1=88THEEND=403
526 POK1=88THEEND=404
527 POK1=88THEEND=405
528 POK1=88THEEND=406
529 POK1=88THEEND=407
530 POK1=88THEEND=408
531 POK1=88THEEND=409
532 POK1=88THEEND=410
533 POK1=88THEEND=411
534 POK1=88THEEND=412
535 POK1=88THEEND=413
536 POK1=88THEEND=414
537 POK1=88THEEND=415
538 POK1=88THEEND=416
539 POK1=88THEEND=417
540 POK1=88THEEND=418
541 POK1=88THEEND=419
542 POK1=88THEEND=420
543 POK1=88THEEND=421
544 POK1=88THEEND=422
545 POK1=88THEEND=423
546 POK1=88THEEND=424
547 POK1=88THEEND=425
548 POK1=88THEEND=426
549 POK1=88THEEND=427
550 POK1=88THEEND=428
551 POK1=88THEEND=429
552 POK1=88THEEND=430
553 POK1=88THEEND=431
554 POK1=88THEEND=432
555 POK1=88THEEND=433
556 POK1=88THEEND=434
557 POK1=88THEEND=435
558 POK1=88THEEND=436
559 POK1=88THEEND=437
560 POK1=88THEEND=438
561 POK1=88THEEND=439
562 POK1=88THEEND=440
563 POK1=88THEEND=441
564 POK1=88THEEND=442
565 POK1=88THEEND=443
566 POK1=88THEEND=444
567 POK1=88THEEND=445
568 POK1=88THEEND=446
569 POK1=88THEEND=447
570 POK1=88THEEND=448
571 POK1=88THEEND=449
572 POK1=88THEEND=450
573 POK1=88THEEND=451
574 POK1=88THEEND=452
575 POK1=88THEEND=453
576 POK1=88THEEND=454
577 POK1=88THEEND=455
578 POK1=88THEEND=456
579 POK1=88THEEND=457
580 POK1=88THEEND=458
581 POK1=88THEEND=459
582 POK1=88THEEND=460
583 POK1=88THEEND=461
584 POK1=88THEEND=462
585 POK1=88THEEND=463
586 POK1=88THEEND=464
587 POK1=88THEEND=465
588 POK1=88THEEND=466
589 POK1=88THEEND=467
590 POK1=88THEEND=468
591 POK1=88THEEND=469
592 POK1=88THEEND=470
593 POK1=88THEEND=471
594 POK1=88THEEND=472
595 POK1=88THEEND=473
596 POK1=88THEEND=474
597 POK1=88THEEND=475
598 POK1=88THEEND=476
599 POK1=88THEEND=477
600 POK1=88THEEND=478
601 POK1=88THEEND=479
602 POK1=88THEEND=480
603 POK1=88THEEND=481
604 POK1=88THEEND=482
605 POK1=88THEEND=483
606 POK1=88THEEND=484
607 POK1=88THEEND=485
608 POK1=88THEEND=486
609 POK1=88THEEND=487
610 POK1=88THEEND=488
611 POK1=88THEEND=489
612 POK1=88THEEND=490
613 POK1=88THEEND=491
614 POK1=88THEEND=492
615 POK1=88THEEND=493
616 POK1=88THEEND=494
617 POK1=88THEEND=495
618 POK1=88THEEND=496
619 POK1=88THEEND=497
620 POK1=88THEEND=498
621 POK1=88THEEND=499
622 POK1=88THEEND=500
623 POK1=88THEEND=501
624 POK1=88THEEND=502
625 POK1=88THEEND=503
626 POK1=88THEEND=504
627 POK1=88THEEND=505
628 POK1=88THEEND=506
629 POK1=88THEEND=507
630 POK1=88THEEND=508
631 POK1=88THEEND=509
632 POK1=88THEEND=510
633 POK1=88THEEND=511
634 POK1=88THEEND=512
635 POK1=88THEEND=513
636 POK1=88THEEND=514
637 POK1=88THEEND=515
638 POK1=88THEEND=516
639 POK1=88THEEND=517
640 POK1=88THEEND=518
641 POK1=88THEEND=519
642 POK1=88THEEND=520
643 POK1=88THEEND=521
644 POK1=88THEEND=522
645 POK1=88THEEND=523
646 POK1=88THEEND=524
647 POK1=88THEEND=525
648 POK1=88THEEND=526
649 POK1=88THEEND=527
650 POK1=88THEEND=528
651 POK1=88THEEND=529
652 POK1=88THEEND=530
653 POK1=88THEEND=531
654 POK1=88THEEND=532
655 POK1=88THEEND=533
656 POK1=88THEEND=534
657 POK1=88THEEND=535
658 POK1=88THEEND=536
659 POK1=88THEEND=537
660 POK1=88THEEND=538
661 POK1=88THEEND=539
662 POK1=88THEEND=540
663 POK1=88THEEND=541
664 POK1=88THEEND=542
665 POK1=88THEEND=543
666 POK1=88THEEND=544
667 POK1=88THEEND=545
668 POK1=88THEEND=546
669 POK1=88THEEND=547
670 POK1=88THEEND=548
671 POK1=88THEEND=549
672 POK1=88THEEND=550
673 POK1=88THEEND=551
674 POK1=88THEEND=552
675 POK1=88THEEND=553
676 POK1=88THEEND=554
677 POK1=88THEEND=555
678 POK1=88THEEND=556
679 POK1=88THEEND=557
680 POK1=88THEEND=558
681 POK1=88THEEND=559
682 POK1=88THEEND=560
683 POK1=88THEEND=561
684 POK1=88THEEND=562
685 POK1=88THEEND=563
686 POK1=88THEEND=564
687 POK1=88THEEND=565
688 POK1=88THEEND=566
689 POK1=88THEEND=567
690 POK1=88THEEND=568
691 POK1=88THEEND=569
692 POK1=88THEEND=570
693 POK1=88THEEND=571
694 POK1=88THEEND=572
695 POK1=88THEEND=573
696 POK1=88THEEND=574
697 POK1=88THEEND=575
698 POK1=88THEEND=576
699 POK1=88THEEND=577
700 POK1=88THEEND=578
701 POK1=88THEEND=579
702 POK1=88THEEND=580
703 POK1=88THEEND=581
704 POK1=88THEEND=582
705 POK1=88THEEND=583
706 POK1=88THEEND=584
707 POK1=88THEEND=585
708 POK1=88THEEND=586
709 POK1=88THEEND=587
710 POK1=88THEEND=588
711 POK1=88THEEND=589
712 POK1=88THEEND=590
713 POK1=88THEEND=591
714 POK1=88THEEND=592
715 POK1=88THEEND=593
716 POK1=88THEEND=594
717 POK1=88THEEND=595
718 POK1=88THEEND=596
719 POK1=88THEEND=597
720 POK1=88THEEND=598
721 POK1=88THEEND=599
722 POK1=88THEEND=600
723 POK1=88THEEND=601
724 POK1=88THEEND=602
725 POK1=88THEEND=603
726 POK1=88THEEND=604
727 POK1=88THEEND=605
728 POK1=88THEEND=606
729 POK1=88THEEND=607
730 POK1=88THEEND=608
731 POK1=88THEEND=609
732 POK1=88THEEND=610
733 POK1=88THEEND=611
734 POK1=88THEEND=612
735 POK1=88THEEND=613
736 POK1=88THEEND=614
737 POK1=88THEEND=615
738 POK1=88THEEND=616
739 POK1=88THEEND=617
740 POK1=88THEEND=618
741 POK1=88THEEND=619
742 POK1=88THEEND=620
743 POK1=88THEEND=621
744 POK1=88THEEND=622
745 POK1=88THEEND=623
746 POK1=88THEEND=624
747 POK1=88THEEND=625
748 POK1=88THEEND=626
749 POK1=88THEEND=627
750 POK1=88THEEND=628
751 POK1=88THEEND=629
752 POK1=88THEEND=630
753 POK1=88THEEND=631
754 POK1=88THEEND=632
755 POK1=88THEEND=633
756 POK1=88THEEND=634
757 POK1=88THEEND=635
758 POK1=88THEEND=636
759 POK1=88THEEND=637
760 POK1=88THEEND=638
761 POK1=88THEEND=639
762 POK1=88THEEND=640
763 POK1=88THEEND=641
764 POK1=88THEEND=642
765 POK1=88THEEND=643
766 POK1=88THEEND=644
767 POK1=88THEEND=645
768 POK1=88THEEND=646
769 POK1=88THEEND=647
770 POK1=88THEEND=648
771 POK1=88THEEND=649
772 POK1=88THEEND=650
773 POK1=88THEEND=651
774 POK1=88THEEND=652
775 POK1=88THEEND=653
776 POK1=88THEEND=654
777 POK1=88THEEND=655
778 POK1=88THEEND=656
779 POK1=88THEEND=657
780 POK1=88THEEND=658
781 POK1=88THEEND=659
782 POK1=88THEEND=660
783 POK1=88THEEND=661
784 POK1=88THEEND=662
785 POK1=88THEEND=663
786 POK1=88THEEND=664
787 POK1=88THEEND=665
788 POK1=88THEEND=666
789 POK1=88THEEND=667
790 POK1=88THEEND=668
791 POK1=88THEEND=669
792 POK1=88THEEND=670
793 POK1=88THEEND=671
794 POK1=88THEEND=672
795 POK1=88THEEND=673
796 POK1=88THEEND=674
797 POK1=88THEEND=675
798 POK1=88THEEND=676
799 POK1=88THEEND=677
800 POK1=88THEEND=678
801 POK1=88THEEND=679
802 POK1=88THEEND=680
803 POK1=88THEEND=681
804 POK1=88THEEND=682
805 POK1=88THEEND=683
806 POK1=88THEEND=684
807 POK1=88THEEND=685
808 POK1=88THEEND=686
809 POK1=88THEEND=
```



CHICANE

This driving game was sent to us by Gareth Rowlands of Cardiff. A width of road between 2 and 7 may be selected (we suggest you start at 7) and the idea is to stay on the road for as long as possible, using the left and right arrow keys to steer. Those of us who were brought up in a more sedentary age may benefit by editing line 100, to adjust the upper limit from 7 to 9. The benefits of selecting easy options do not last long - the road narrows as you progress.

```

10 REM ***CHICANE*** G. ROWLANDS
10 CLS:PRINT"ROAD WIDTH: 5-10"
20 AUDIO(OFF):MOTOR(OFF):M1=-
"ROAD:CHICANE:CODE:0000":GOTO 100
30 FOR I=19980 TO P-L0:POKE 1256:NEXT
40 FOR I=P-1 TO P-10:D=EDPOKE 1156:NEXT
50 FOR I=P-1+80 TO 3999:POKE 1256:NEXT:RETURN
60 FOR I=10:NEXT:D=10:DEC256:NEXT:TIMER=0
70 FOR Q=1 TO 10
80 FOR I=1 TO 10:IF R=PROG2:IF R=1 THEN LC=15:PC=299
ELSE LC=255:PC=179
90 FOR C=PC-(PDED200-320)-(PROG200-320):
POKEP=LC,LC:POKEP=EDPOKE:P=I+2004:AND R=1:
I=I-10:AND R=1:DEC256:IF PROG2=1 THEN
100 ELSE POKEP,38:NEXT
110 IF P=2999 THEN D=38 ELSE D=1
120 PRINT@D,INT(TIMER+100/1000):
:PRINT@D+13,"MILES":SCREEN 8:NEXT
130 A=A+14:21:SCREEN 8
140 PRINT@D+14,"ROAD":PRINT@D+15,"NARROWS":
150 LD=PRO+RD=A:GOSUB 218:SCREEN 8:NEXT
160 T=TIMER:FOR I=1 TO 10:DEC256:
:POKEP=RD=2,28:POKEP=RD=0,28:
:POKEP=RD=2,28:POKEP=RD=0,28:NEXT
170 END=1:THEN T=0
180 PLAY M1:PRINT@D,"YOU
 SURVIVED":PRINT@D,"MILES":
190 PRINT@D,"WIDTH OF ROAD":SCREEN 1
200 A=1:INKEY$=VAL$H-A:IF VAL$H=1 THEN 190
210 CUBUS=RD=RA=VAL$H-A:
:GOSUB 218
220 P=2992:OP=15:GOSUB 218:GOTO 90
230 B=B-1:LD=LD+1:IF B=1 THEN RETURN
240 B=B-1:RD=RD+1:IF B<1 THEN RETURN ELSE 230
250 P=P+1:READ B$IF B$="END":THEN RETURN
260 POKE P,2998:VAL$H="":POKE 2070 299
270 DATA 100:RETURN:END
280 DATA 96:DEC256:END
290 DATA 97,95,90,98:END
300 DATA 98,99,91,99:END
310 DATA 94,95,97,98:END
320 DATA 95,96,98,99:END

```

The machine code (lines 298 to 399) controls the down-scroll. The following listing may be useful.

	Machine Code
298	0000
299	0000
300	00 00
301	00 00
302	0000
303	0000
304	0000
305	0000
306	0000
307	0000
308	0000 00
309	0000
310	00 00
311	0000
312	0000 00
313	0000
314	0000
315	0000
316	0000
317	0000
318	0000
319	0000
320	0000
321	0000
322	0000
323	0000
324	0000
325	0000
326	0000
327	0000
328	0000
329	0000
330	0000
331	0000
332	0000
333	0000
334	0000
335	0000
336	0000
337	0000
338	0000
339	0000
340	0000
341	0000
342	0000
343	0000
344	0000
345	0000
346	0000
347	0000
348	0000
349	0000
350	0000
351	0000
352	0000
353	0000
354	0000
355	0000
356	0000
357	0000
358	0000
359	0000
360	0000
361	0000
362	0000
363	0000
364	0000
365	0000
366	0000
367	0000
368	0000
369	0000
370	0000
371	0000
372	0000
373	0000
374	0000
375	0000
376	0000
377	0000
378	0000
379	0000
380	0000
381	0000
382	0000
383	0000
384	0000
385	0000
386	0000
387	0000
388	0000
389	0000
390	0000
391	0000
392	0000
393	0000
394	0000
395	0000
396	0000
397	0000
398	0000
399	0000

CODEBREAKER

This is a short program to further illustrate the power of Dragon's string handling capabilities. It translates a typed message into a code which is constructed by randomly permuting the allowed letters. Because the random permutation has been generated using PRNG(1-10) to initialise the random sequence, the code can be deciphered using the same program providing the transmission code is known.

Of course, given time all such codes can be 'cracked' and one of the ways to do that is to use the knowledge gained from 'Letter Count' concerning the frequencies of each of the letters of the alphabet. For a literary reference read 'THE GOLD BUG' in Edgar Allan Poe's 'Tales of Mystery and Suspense'.

```

1 REM CODEBREAKER A.J.S BYTES MARCH
10 CLS:CLEAR#1#CLS=-
"CODEBREAKER:CODE:0000":GOTO 100
20 INPUT#1 FOR CODING,2 FOR DECODING:
40 ON C GOSUB 601:STEP
50 R=PRNG(100:PRINT"OUR TRANSMISSION
 CODE IS":R
60 GOSUB 601
70 PRINT@R,"NOW KEY IN YOUR MESSAGE"
80 PRINT#1=250 CHARACTERS LONG AND ENDING
 WITH :ENTER>":"
90 GOSUB 601:GOSUB 601:CLS
100 PRINT@1,"YOUR MESSAGE READS":
GOSUB 601:PRINT@1,LIN:RETURN
110 INPUT#2# THE TRANSMISSION CODE#A
120 LD=2:RD=0:FOR I=1 TO R-2:RD=RD+1:STEP -1
130 C01=CHR#I#I=39 TO 1 STEP -1
140 I=I-PRNG(100:LEFT#I,1)=I-1:RIGHT#I,9-I)=I
+PRNG(100:LEFT#I,1)=I-1:RIGHT#I,9-I)=I
150 LD=LD+1:IF LD=R THEN RETURN
160 LD=LD+1:IF LD=R THEN RETURN
170 LD=LD+1:IF LD=R THEN RETURN
180 LD=LD+1:IF LD=R THEN RETURN
190 LD=LD+1:IF LD=R THEN RETURN
200 LD=LD+1:IF LD=R THEN RETURN
210 LD=LD+1:IF LD=R THEN RETURN
220 LD=LD+1:IF LD=R THEN RETURN
230 LD=LD+1:IF LD=R THEN RETURN
240 LD=LD+1:IF LD=R THEN RETURN
250 LD=LD+1:IF LD=R THEN RETURN
260 LD=LD+1:IF LD=R THEN RETURN
270 LD=LD+1:IF LD=R THEN RETURN
280 LD=LD+1:IF LD=R THEN RETURN
290 LD=LD+1:IF LD=R THEN RETURN
300 LD=LD+1:IF LD=R THEN RETURN
310 LD=LD+1:IF LD=R THEN RETURN
320 LD=LD+1:IF LD=R THEN RETURN
330 LD=LD+1:IF LD=R THEN RETURN
340 LD=LD+1:IF LD=R THEN RETURN
350 LD=LD+1:IF LD=R THEN RETURN
360 LD=LD+1:IF LD=R THEN RETURN
370 LD=LD+1:IF LD=R THEN RETURN
380 LD=LD+1:IF LD=R THEN RETURN
390 LD=LD+1:IF LD=R THEN RETURN
400 LD=LD+1:IF LD=R THEN RETURN
410 LD=LD+1:IF LD=R THEN RETURN
420 LD=LD+1:IF LD=R THEN RETURN
430 LD=LD+1:IF LD=R THEN RETURN
440 LD=LD+1:IF LD=R THEN RETURN
450 LD=LD+1:IF LD=R THEN RETURN
460 LD=LD+1:IF LD=R THEN RETURN
470 LD=LD+1:IF LD=R THEN RETURN
480 LD=LD+1:IF LD=R THEN RETURN
490 LD=LD+1:IF LD=R THEN RETURN
500 LD=LD+1:IF LD=R THEN RETURN
510 LD=LD+1:IF LD=R THEN RETURN
520 LD=LD+1:IF LD=R THEN RETURN
530 LD=LD+1:IF LD=R THEN RETURN
540 LD=LD+1:IF LD=R THEN RETURN
550 LD=LD+1:IF LD=R THEN RETURN
560 LD=LD+1:IF LD=R THEN RETURN
570 LD=LD+1:IF LD=R THEN RETURN
580 LD=LD+1:IF LD=R THEN RETURN
590 LD=LD+1:IF LD=R THEN RETURN
600 LD=LD+1:IF LD=R THEN RETURN
610 LD=LD+1:IF LD=R THEN RETURN
620 LD=LD+1:IF LD=R THEN RETURN
630 LD=LD+1:IF LD=R THEN RETURN
640 LD=LD+1:IF LD=R THEN RETURN
650 LD=LD+1:IF LD=R THEN RETURN
660 LD=LD+1:IF LD=R THEN RETURN
670 LD=LD+1:IF LD=R THEN RETURN
680 LD=LD+1:IF LD=R THEN RETURN
690 LD=LD+1:IF LD=R THEN RETURN
700 LD=LD+1:IF LD=R THEN RETURN
710 LD=LD+1:IF LD=R THEN RETURN
720 LD=LD+1:IF LD=R THEN RETURN
730 LD=LD+1:IF LD=R THEN RETURN
740 LD=LD+1:IF LD=R THEN RETURN
750 LD=LD+1:IF LD=R THEN RETURN
760 LD=LD+1:IF LD=R THEN RETURN
770 LD=LD+1:IF LD=R THEN RETURN
780 LD=LD+1:IF LD=R THEN RETURN
790 LD=LD+1:IF LD=R THEN RETURN
800 LD=LD+1:IF LD=R THEN RETURN
810 LD=LD+1:IF LD=R THEN RETURN
820 LD=LD+1:IF LD=R THEN RETURN
830 LD=LD+1:IF LD=R THEN RETURN
840 LD=LD+1:IF LD=R THEN RETURN
850 LD=LD+1:IF LD=R THEN RETURN
860 LD=LD+1:IF LD=R THEN RETURN
870 LD=LD+1:IF LD=R THEN RETURN
880 LD=LD+1:IF LD=R THEN RETURN
890 LD=LD+1:IF LD=R THEN RETURN
900 LD=LD+1:IF LD=R THEN RETURN
910 LD=LD+1:IF LD=R THEN RETURN
920 LD=LD+1:IF LD=R THEN RETURN
930 LD=LD+1:IF LD=R THEN RETURN
940 LD=LD+1:IF LD=R THEN RETURN
950 LD=LD+1:IF LD=R THEN RETURN
960 LD=LD+1:IF LD=R THEN RETURN
970 LD=LD+1:IF LD=R THEN RETURN
980 LD=LD+1:IF LD=R THEN RETURN
990 LD=LD+1:IF LD=R THEN RETURN
1000 LD=LD+1:IF LD=R THEN RETURN
1010 LD=LD+1:IF LD=R THEN RETURN
1020 LD=LD+1:IF LD=R THEN RETURN
1030 LD=LD+1:IF LD=R THEN RETURN
1040 LD=LD+1:IF LD=R THEN RETURN
1050 LD=LD+1:IF LD=R THEN RETURN
1060 LD=LD+1:IF LD=R THEN RETURN
1070 LD=LD+1:IF LD=R THEN RETURN
1080 LD=LD+1:IF LD=R THEN RETURN
1090 LD=LD+1:IF LD=R THEN RETURN
1100 LD=LD+1:IF LD=R THEN RETURN
1110 LD=LD+1:IF LD=R THEN RETURN
1120 LD=LD+1:IF LD=R THEN RETURN
1130 LD=LD+1:IF LD=R THEN RETURN
1140 LD=LD+1:IF LD=R THEN RETURN
1150 LD=LD+1:IF LD=R THEN RETURN
1160 LD=LD+1:IF LD=R THEN RETURN
1170 LD=LD+1:IF LD=R THEN RETURN
1180 LD=LD+1:IF LD=R THEN RETURN
1190 LD=LD+1:IF LD=R THEN RETURN
1200 LD=LD+1:IF LD=R THEN RETURN
1210 LD=LD+1:IF LD=R THEN RETURN
1220 LD=LD+1:IF LD=R THEN RETURN
1230 LD=LD+1:IF LD=R THEN RETURN
1240 LD=LD+1:IF LD=R THEN RETURN
1250 LD=LD+1:IF LD=R THEN RETURN
1260 LD=LD+1:IF LD=R THEN RETURN
1270 LD=LD+1:IF LD=R THEN RETURN
1280 LD=LD+1:IF LD=R THEN RETURN
1290 LD=LD+1:IF LD=R THEN RETURN
1300 LD=LD+1:IF LD=R THEN RETURN
1310 LD=LD+1:IF LD=R THEN RETURN
1320 LD=LD+1:IF LD=R THEN RETURN
1330 LD=LD+1:IF LD=R THEN RETURN
1340 LD=LD+1:IF LD=R THEN RETURN
1350 LD=LD+1:IF LD=R THEN RETURN
1360 LD=LD+1:IF LD=R THEN RETURN
1370 LD=LD+1:IF LD=R THEN RETURN
1380 LD=LD+1:IF LD=R THEN RETURN
1390 LD=LD+1:IF LD=R THEN RETURN
1400 LD=LD+1:IF LD=R THEN RETURN
1410 LD=LD+1:IF LD=R THEN RETURN
1420 LD=LD+1:IF LD=R THEN RETURN
1430 LD=LD+1:IF LD=R THEN RETURN
1440 LD=LD+1:IF LD=R THEN RETURN
1450 LD=LD+1:IF LD=R THEN RETURN
1460 LD=LD+1:IF LD=R THEN RETURN
1470 LD=LD+1:IF LD=R THEN RETURN
1480 LD=LD+1:IF LD=R THEN RETURN
1490 LD=LD+1:IF LD=R THEN RETURN
1500 LD=LD+1:IF LD=R THEN RETURN
1510 LD=LD+1:IF LD=R THEN RETURN
1520 LD=LD+1:IF LD=R THEN RETURN
1530 LD=LD+1:IF LD=R THEN RETURN
1540 LD=LD+1:IF LD=R THEN RETURN
1550 LD=LD+1:IF LD=R THEN RETURN
1560 LD=LD+1:IF LD=R THEN RETURN
1570 LD=LD+1:IF LD=R THEN RETURN
1580 LD=LD+1:IF LD=R THEN RETURN
1590 LD=LD+1:IF LD=R THEN RETURN
1600 LD=LD+1:IF LD=R THEN RETURN
1610 LD=LD+1:IF LD=R THEN RETURN
1620 LD=LD+1:IF LD=R THEN RETURN
1630 LD=LD+1:IF LD=R THEN RETURN
1640 LD=LD+1:IF LD=R THEN RETURN
1650 LD=LD+1:IF LD=R THEN RETURN
1660 LD=LD+1:IF LD=R THEN RETURN
1670 LD=LD+1:IF LD=R THEN RETURN
1680 LD=LD+1:IF LD=R THEN RETURN
1690 LD=LD+1:IF LD=R THEN RETURN
1700 LD=LD+1:IF LD=R THEN RETURN
1710 LD=LD+1:IF LD=R THEN RETURN
1720 LD=LD+1:IF LD=R THEN RETURN
1730 LD=LD+1:IF LD=R THEN RETURN
1740 LD=LD+1:IF LD=R THEN RETURN
1750 LD=LD+1:IF LD=R THEN RETURN
1760 LD=LD+1:IF LD=R THEN RETURN
1770 LD=LD+1:IF LD=R THEN RETURN
1780 LD=LD+1:IF LD=R THEN RETURN
1790 LD=LD+1:IF LD=R THEN RETURN
1800 LD=LD+1:IF LD=R THEN RETURN
1810 LD=LD+1:IF LD=R THEN RETURN
1820 LD=LD+1:IF LD=R THEN RETURN
1830 LD=LD+1:IF LD=R THEN RETURN
1840 LD=LD+1:IF LD=R THEN RETURN
1850 LD=LD+1:IF LD=R THEN RETURN
1860 LD=LD+1:IF LD=R THEN RETURN
1870 LD=LD+1:IF LD=R THEN RETURN
1880 LD=LD+1:IF LD=R THEN RETURN
1890 LD=LD+1:IF LD=R THEN RETURN
1900 LD=LD+1:IF LD=R THEN RETURN
1910 LD=LD+1:IF LD=R THEN RETURN
1920 LD=LD+1:IF LD=R THEN RETURN
1930 LD=LD+1:IF LD=R THEN RETURN
1940 LD=LD+1:IF LD=R THEN RETURN
1950 LD=LD+1:IF LD=R THEN RETURN
1960 LD=LD+1:IF LD=R THEN RETURN
1970 LD=LD+1:IF LD=R THEN RETURN
1980 LD=LD+1:IF LD=R THEN RETURN
1990 LD=LD+1:IF LD=R THEN RETURN
2000 LD=LD+1:IF LD=R THEN RETURN
2010 LD=LD+1:IF LD=R THEN RETURN
2020 LD=LD+1:IF LD=R THEN RETURN
2030 LD=LD+1:IF LD=R THEN RETURN
2040 LD=LD+1:IF LD=R THEN RETURN
2050 LD=LD+1:IF LD=R THEN RETURN
2060 LD=LD+1:IF LD=R THEN RETURN
2070 LD=LD+1:IF LD=R THEN RETURN
2080 LD=LD+1:IF LD=R THEN RETURN
2090 LD=LD+1:IF LD=R THEN RETURN
2100 LD=LD+1:IF LD=R THEN RETURN
2110 LD=LD+1:IF LD=R THEN RETURN
2120 LD=LD+1:IF LD=R THEN RETURN
2130 LD=LD+1:IF LD=R THEN RETURN
2140 LD=LD+1:IF LD=R THEN RETURN
2150 LD=LD+1:IF LD=R THEN RETURN
2160 LD=LD+1:IF LD=R THEN RETURN
2170 LD=LD+1:IF LD=R THEN RETURN
2180 LD=LD+1:IF LD=R THEN RETURN
2190 LD=LD+1:IF LD=R THEN RETURN
2200 LD=LD+1:IF LD=R THEN RETURN
2210 LD=LD+1:IF LD=R THEN RETURN
2220 LD=LD+1:IF LD=R THEN RETURN
2230 LD=LD+1:IF LD=R THEN RETURN
2240 LD=LD+1:IF LD=R THEN RETURN
2250 LD=LD+1:IF LD=R THEN RETURN
2260 LD=LD+1:IF LD=R THEN RETURN
2270 LD=LD+1:IF LD=R THEN RETURN
2280 LD=LD+1:IF LD=R THEN RETURN
2290 LD=LD+1:IF LD=R THEN RETURN
2300 LD=LD+1:IF LD=R THEN RETURN
2310 LD=LD+1:IF LD=R THEN RETURN
2320 LD=LD+1:IF LD=R THEN RETURN
2330 LD=LD+1:IF LD=R THEN RETURN
2340 LD=LD+1:IF LD=R THEN RETURN
2350 LD=LD+1:IF LD=R THEN RETURN
2360 LD=LD+1:IF LD=R THEN RETURN
2370 LD=LD+1:IF LD=R THEN RETURN
2380 LD=LD+1:IF LD=R THEN RETURN
2390 LD=LD+1:IF LD=R THEN RETURN
2400 LD=LD+1:IF LD=R THEN RETURN
2410 LD=LD+1:IF LD=R THEN RETURN
2420 LD=LD+1:IF LD=R THEN RETURN
2430 LD=LD+1:IF LD=R THEN RETURN
2440 LD=LD+1:IF LD=R THEN RETURN
2450 LD=LD+1:IF LD=R THEN RETURN
2460 LD=LD+1:IF LD=R THEN RETURN
2470 LD=LD+1:IF LD=R THEN RETURN
2480 LD=LD+1:IF LD=R THEN RETURN
2490 LD=LD+1:IF LD=R THEN RETURN
2500 LD=LD+1:IF LD=R THEN RETURN
2510 LD=LD+1:IF LD=R THEN RETURN
2520 LD=LD+1:IF LD=R THEN RETURN
2530 LD=LD+1:IF LD=R THEN RETURN
2540 LD=LD+1:IF LD=R THEN RETURN
2550 LD=LD+1:IF LD=R THEN RETURN
2560 LD=LD+1:IF LD=R THEN RETURN
2570 LD=LD+1:IF LD=R THEN RETURN
2580 LD=LD+1:IF LD=R THEN RETURN
2590 LD=LD+1:IF LD=R THEN RETURN
2600 LD=LD+1:IF LD=R THEN RETURN
2610 LD=LD+1:IF LD=R THEN RETURN
2620 LD=LD+1:IF LD=R THEN RETURN
2630 LD=LD+1:IF LD=R THEN RETURN
2640 LD=LD+1:IF LD=R THEN RETURN
2650 LD=LD+1:IF LD=R THEN RETURN
2660 LD=LD+1:IF LD=R THEN RETURN
2670 LD=LD+1:IF LD=R THEN RETURN
2680 LD=LD+1:IF LD=R THEN RETURN
2690 LD=LD+1:IF LD=R THEN RETURN
2700 LD=LD+1:IF LD=R THEN RETURN
2710 LD=LD+1:IF LD=R THEN RETURN
2720 LD=LD+1:IF LD=R THEN RETURN
2730 LD=LD+1:IF LD=R THEN RETURN
2740 LD=LD+1:IF LD=R THEN RETURN
2750 LD=LD+1:IF LD=R THEN RETURN
2760 LD=LD+1:IF LD=R THEN RETURN
2770 LD=LD+1:IF LD=R THEN RETURN
2780 LD=LD+1:IF LD=R THEN RETURN
2790 LD=LD+1:IF LD=R THEN RETURN
2800 LD=LD+1:IF LD=R THEN RETURN
2810 LD=LD+1:IF LD=R THEN RETURN
2820 LD=LD+1:IF LD=R THEN RETURN
2830 LD=LD+1:IF LD=R THEN RETURN
2840 LD=LD+1:IF LD=R THEN RETURN
2850 LD=LD+1:IF LD=R THEN RETURN
2860 LD=LD+1:IF LD=R THEN RETURN
2870 LD=LD+1:IF LD=R THEN RETURN
2880 LD=LD+1:IF LD=R THEN RETURN
2890 LD=LD+1:IF LD=R THEN RETURN
2900 LD=LD+1:IF LD=R THEN RETURN
2910 LD=LD+1:IF LD=R THEN RETURN
2920 LD=LD+1:IF LD=R THEN RETURN
2930 LD=LD+1:IF LD=R THEN RETURN
2940 LD=LD+1:IF LD=R THEN RETURN
2950 LD=LD+1:IF LD=R THEN RETURN
2960 LD=LD+1:IF LD=R THEN RETURN
2970 LD=LD+1:IF LD=R THEN RETURN
2980 LD=LD+1:IF LD=R THEN RETURN
2990 LD=LD+1:IF LD=R THEN RETURN
3000 LD=LD+1:IF LD=R THEN RETURN
3010 LD=LD+1:IF LD=R THEN RETURN
3020 LD=LD+1:IF LD=R THEN RETURN
3030 LD=LD+1:IF LD=R THEN RETURN
3040 LD=LD+1:IF LD=R THEN RETURN
3050 LD=LD+1:IF LD=R THEN RETURN
3060 LD=LD+1:IF LD=R THEN RETURN
3070 LD=LD+1:IF LD=R THEN RETURN
3080 LD=LD+1:IF LD=R THEN RETURN
3090 LD=LD+1:IF LD=R THEN RETURN
3100 LD=LD+1:IF LD=R THEN RETURN
3110 LD=LD+1:IF LD=R THEN RETURN
3120 LD=LD+1:IF LD=R THEN RETURN
3130 LD=LD+1:IF LD=R THEN RETURN
3140 LD=LD+1:IF LD=R THEN RETURN
3150 LD=LD+1:IF LD=R THEN RETURN
3160 LD=LD+1:IF LD=R THEN RETURN
3170 LD=LD+1:IF LD=R THEN RETURN
3180 LD=LD+1:IF LD=R THEN RETURN
3190 LD=LD+1:IF LD=R THEN RETURN
3200 LD=LD+1:IF LD=R THEN RETURN
3210 LD=LD+1:IF LD=R THEN RETURN
3220 LD=LD+1:IF LD=R THEN RETURN
3230 LD=LD+1:IF LD=R THEN RETURN
3240 LD=LD+1:IF LD=R THEN RETURN
3250 LD=LD+1:IF LD=R THEN RETURN
3260 LD=LD+1:IF LD=R THEN RETURN
3270 LD=LD+1:IF LD=R THEN RETURN
3280 LD=LD+1:IF LD=R THEN RETURN
3290 LD=LD+1:IF LD=R THEN RETURN
3300 LD=LD+1:IF LD=R THEN RETURN
3310 LD=LD+1:IF LD=R THEN RETURN
3320 LD=LD+1:IF LD=R THEN RETURN
3330 LD=LD+1:IF LD=R THEN RETURN
3340 LD=LD+1:IF LD=R THEN RETURN
3350 LD=LD+1:IF LD=R THEN RETURN
3360 LD=LD+1:IF LD=R THEN RETURN
3370 LD=LD+1:IF LD=R THEN RETURN
3380 LD=LD+1:IF LD=R THEN RETURN
3390 LD=LD+1:IF LD=R THEN RETURN
3400 LD=LD+1:IF LD=R THEN RETURN
3410 LD=LD+1:IF LD=R THEN RETURN
3420 LD=LD+1:IF LD=R THEN RETURN
3430 LD=LD+1:IF LD=R THEN RETURN
3440 LD=LD+1:IF LD=R THEN RETURN
3450 LD=LD+1:IF LD=R THEN RETURN
3460 LD=LD+1:IF LD=R THEN RETURN
3470 LD=LD+1:IF LD=R THEN RETURN
3480 LD=LD+1:IF LD=R THEN RETURN
3490 LD=LD+1:IF LD=R THEN RETURN
3500 LD=LD+1:IF LD=R THEN RETURN
3510 LD=LD+1:IF LD=R THEN RETURN
3520 LD=LD+1:IF LD=R THEN RETURN
3530 LD=LD+1:IF LD=R THEN RETURN
3540 LD=LD+1:IF LD=R THEN RETURN
3550 LD=LD+1:IF LD=R THEN RETURN
3560 LD=LD+1:IF LD=R THEN RETURN
3570 LD=LD+1:IF LD=R THEN RETURN
3580 LD=LD+1:IF LD=R THEN RETURN
3590 LD=LD+1:IF LD=R THEN RETURN
3600 LD=LD+1:IF LD=R THEN RETURN
3610 LD=LD+1:IF LD=R THEN RETURN
3620 LD=LD+1:IF LD=R THEN RETURN
3630 LD=LD+1:IF LD=R THEN RETURN
3640 LD=LD+1:IF LD=R THEN RETURN
3650 LD=LD+1:IF LD=R THEN RETURN
3660 LD=LD+1:IF LD=R THEN RETURN
3670 LD=LD+1:IF LD=R THEN RETURN
3680 LD=LD+1:IF LD=R THEN RETURN
3690 LD=LD+1:IF LD=R THEN RETURN
3700 LD=LD+1:IF LD=R THEN RETURN
3710 LD=LD+1:IF LD=R THEN RETURN
3720 LD=LD+1:IF LD=R THEN RETURN
3730 LD=LD+1:IF LD=R THEN RETURN
3740 LD=LD+1:IF LD=R THEN RETURN
3750 LD=LD+1:IF LD=R THEN RETURN
3760 LD=LD+1:IF LD=R THEN RETURN
3770 LD=LD+1:IF LD=R THEN RETURN
3780 LD=LD+1:IF LD=R THEN RETURN
3790 LD=LD+1:IF LD=R THEN RETURN
3800 LD=LD+1:IF LD=R THEN RETURN
3810 LD=LD+1:IF LD=R THEN RETURN
3820 LD=LD+1:IF LD=R THEN RETURN
3830 LD=LD+1:IF LD=R THEN RETURN
3840 LD=LD+1:IF LD=R THEN RETURN
3850 LD=LD+1:IF LD=R THEN RETURN
3860 LD=LD+1:IF LD=R THEN RETURN
3870 LD=LD+1:IF LD=R THEN RETURN
3880 LD=LD+1:IF LD=R THEN RETURN
3890 LD=LD+1:IF LD=R THEN RETURN
3900 LD=LD+1:IF LD=R THEN RETURN
3910 LD=LD+1:IF LD=R THEN RETURN
3920 LD=LD+1:IF LD=R THEN RETURN
3930 LD=LD+1:IF LD=R THEN RETURN
3940 LD=LD+1:IF LD=R THEN RETURN
3950 LD=LD+1:IF LD=R THEN RETURN
3960 LD=LD+1:IF LD=R THEN RETURN
3970 LD=LD+1:IF LD=R THEN RETURN
3980 LD=LD+1:IF LD=R THEN RETURN
3990 LD=LD+1:IF LD=R THEN RETURN
4000 LD=LD+1:IF LD=R THEN RETURN
4010 LD=LD+1:IF LD=R THEN RETURN
4020 LD=LD+1:IF LD=R THEN RETURN
4030 LD=LD+1:IF LD=R THEN RETURN
4040 LD=LD+1:IF LD=R THEN RETURN
4050 LD=LD+1:IF LD=R THEN RETURN
4060 LD=LD+1:IF LD=R THEN RETURN
4070 LD=LD+1:IF LD=R THEN RETURN
4080 LD=LD+1:IF LD=R THEN RETURN
4090 LD=LD+1:IF LD=R THEN RETURN
4100 LD=LD+1:IF LD=R THEN RETURN
4110 LD=LD+1:IF LD=R THEN RETURN
4120 LD=LD+1:IF LD=R THEN RETURN
4130 LD=LD+1:IF LD=R THEN RETURN
4140 LD=LD+1:IF LD=R THEN RETURN
4150 LD=LD+1:IF LD=R THEN RETURN
4160 LD=LD+1:IF LD=R THEN RETURN
4170 LD=LD+1:IF LD=R THEN RETURN
4180 LD=LD+1:IF LD=R THEN RETURN
4190 LD=LD+1:IF LD=R THEN RETURN
4200 LD=LD+1:IF LD=R THEN RETURN
4210 LD=LD+1:IF LD=R THEN RETURN
4220 LD=LD+1:IF LD=R THEN RETURN
4230 LD=LD+1:IF LD=R THEN RETURN
4240 LD=LD+1:IF LD=R THEN RETURN
4250 LD=LD+1:IF LD=R THEN RETURN
4260 LD=LD+1:IF LD=R THEN RETURN
4270 LD=LD+1:IF LD=R THEN RETURN
4280 LD=LD+1:IF LD=R THEN RETURN
4290 LD=LD+1:IF LD=R THEN RETURN
4300 LD=LD+1:IF LD=R THEN RETURN
4310 LD=LD+1:IF LD=R THEN RETURN
4320 LD=LD+1:IF LD=R THEN RETURN
4330 LD=LD+1:IF LD=R THEN RETURN
4340 LD=LD+1:IF LD=R THEN RETURN
4350 LD=LD+1:IF LD=R THEN RETURN
4360 LD=LD+1:IF LD=R THEN RETURN
4370 LD=LD+1:IF LD=R THEN RETURN
4380 LD=LD+1:IF LD=R THEN RETURN
4390 LD=LD+1:IF LD=R THEN RETURN
4400 LD=LD+1:IF LD=R THEN RETURN
4410 LD=LD+1:IF LD=R THEN RETURN
4420 LD=LD+1:IF LD=R THEN RETURN
4430 LD=LD+1:IF LD=R THEN RETURN
4440 LD=LD+1:IF LD=R THEN RETURN
4450 LD=LD+1:IF LD=R THEN RETURN
4460 LD=LD+1:IF LD=R THEN RETURN
4470 LD=LD+1:IF LD=R THEN RETURN
4480 LD=LD+1:IF LD=R THEN RETURN
4490 LD=LD+1:IF LD=R THEN RETURN
4500 LD=LD+1:IF LD=R THEN RETURN
4510 LD=LD+1:IF LD=R THEN RETURN
4520 LD=LD+1:IF LD=R THEN RETURN
4530 LD=LD+1:IF LD=R THEN RETURN
4540 LD=LD+1:IF LD=R THEN RETURN
4550 LD=LD+1:IF LD=R THEN RETURN
4560 LD=LD+1:IF LD=R THEN RETURN
4570 LD=LD+1:IF LD=R THEN RETURN
4580 LD=LD+1:IF LD=R THEN RETURN
4590 LD=LD+1:IF LD=R THEN RETURN
4600 LD=LD+1:IF LD=R THEN RETURN
4610 LD=LD+1:IF LD=R THEN RETURN
4620 LD=LD+1:IF LD=R THEN RETURN
4630 LD=LD+1:IF LD=R THEN RETURN
4640 LD=LD+1:IF LD=R THEN RETURN
4650 LD=LD+1:IF LD=R THEN RETURN
4660 LD=LD+1:IF LD=R THEN RETURN
4670 LD=LD+1:IF LD=R THEN RETURN
4680 LD=LD+1:IF LD=R THEN RETURN
4690 LD=LD+1:IF LD=R THEN RETURN
4700 LD=LD+1:IF LD=R THEN RETURN
4710 LD=LD+1:IF LD=R THEN RETURN
4720 LD=LD+1:IF LD=R THEN RETURN
4730 LD=LD+1:IF LD=R THEN RETURN
4740 LD=LD+1:IF LD=R THEN RETURN
4750 LD=LD+1:IF LD=R THEN RETURN
4760 LD=LD+1:IF LD=R THEN RETURN
4770 LD=LD+1:IF LD=R THEN RETURN
4780 LD=LD+1:IF LD=R THEN RETURN
4790 LD=LD+1:IF LD=R THEN RETURN
4800 LD=LD+1:IF LD=R THEN RETURN
4810 LD=LD+1:IF LD=R THEN RETURN
4820 LD=LD+1:IF LD=R THEN RETURN
4830 LD=LD+1:IF LD=R THEN RETURN
4840 LD=LD+1:IF LD=R THEN RETURN
4850 LD=LD+1:IF LD=R THEN RETURN
4860 LD=LD+1:IF LD=R THEN RETURN
4870 LD=LD+1:IF LD=R THEN RETURN
4880 LD=LD+1:IF LD=R THEN RETURN
4890 LD=LD+1:IF LD=R THEN RETURN
4900 LD=LD+1:IF LD=R THEN RETURN
4910 LD=LD+1:IF LD=R THEN RETURN
4920 LD=LD+1:IF LD=R THEN RETURN
4930 LD=LD+1:IF LD=R THEN RETURN
4940 LD=LD+1:IF LD=R THEN RETURN
4950 LD=LD+1:IF LD=R THEN RETURN
4960 LD=LD+1:IF LD=R THEN RETURN
4970 LD=LD+1:IF LD=R THEN RETURN
4980 LD=LD+1:IF LD=R THEN RETURN
4990 LD=LD+1:IF LD=R THEN RETURN
5000 LD=LD+1:IF LD=R THEN RETURN
5010 LD=LD+1:IF LD=R THEN RETURN
5020 LD=LD+1:IF LD=R THEN RETURN
5030 LD=LD+1:IF LD=R THEN RETURN
5040 LD=LD+1:IF LD=R THEN RETURN
5050 LD=LD+1:IF LD=R THEN RETURN
5060 LD=LD+1:IF LD=R THEN RETURN
5070 LD=LD+1:IF LD=R THEN RETURN
5080 LD=LD+1:IF LD=R THEN RETURN
5090 LD=LD+1:IF LD=R THEN RETURN
5100 LD=LD+1:IF LD=R THEN RETURN
5110 LD=LD+1:IF LD=R THEN RETURN
5120 LD=LD+1:IF LD=R THEN RETURN
5130 LD=LD+1:IF LD=R THEN RETURN
5140 LD=LD+1:IF LD=R THEN RETURN
5150 LD=LD+1:IF LD=R THEN RETURN
5160 LD=LD+1:IF LD=R THEN RETURN
5170 LD=LD+1:IF LD=R THEN RETURN
5180 LD=LD+1:IF LD=R THEN RETURN
5190 LD=LD+1:IF LD=R THEN RETURN
5200 LD=LD+1:IF LD=R THEN RETURN
5210 LD=LD+1:IF LD=R THEN RETURN
5220 LD=LD+1:IF LD=R THEN RETURN
5230 LD=LD+1:IF LD=R THEN RETURN
5240 LD=LD+1:IF LD=R THEN RETURN
5250 LD=LD+1:IF LD=R THEN RETURN
5260 LD=LD+1:IF LD=R THEN RETURN
5270 LD=LD+1:IF LD=R THEN RETURN
5280 LD=LD+1:IF LD=R THEN RETURN
5290 LD=LD+1:IF LD=R THEN RETURN
53
```

LETTER COUNT



This program counts the number of occurrences of letters in writing. Just type in the words and watch the bar chart build up. There is no backspace so type carefully. The program stops when any bar reaches the top. Then you can press **ENTER** for a table of the number of occurrences of each letter. You can press **ENTER** for the table at any time during a run but the program is then terminated.

Another way to display the bar chart requires the following changed lines, with line 100 deleted:

10. CLUSTERING

TAKE 1000
LINES



The **LIN**E command can draw lines on the screen in any direction. Unlike the **DRWILL** command **H** is not limited to eight directions. The command receives the co-ordinates of the end points of

the line. It is very useful to have the high-resolution grid for reference. Once the end points have been established the LINE command takes another parameter, PSSET or PRESET. PSSET draws the line in the foreground colour whereas PRESET draws it in the background colour. The colours can be chosen using the COLOR command. This has two parameters FG, the foreground colour and BG, the background. The colours must be chosen with respect to the SCREEN in use. If the COLOR command is omitted the foreground colour will be the highest number available on that screen and the background the lowest. There is another form of the command LINE-(X1,Y1,X2,Y2) which draws a line between the new point and the last one used in a LINE command. If the first LINE command in the program is of this form, a line is drawn from the centre of the screen to the point. Using PRESET, the first time round might be one way of removing this line. In addition to PSSET and PRESET there is another parameter: by adding B to the LINE command a box is drawn instead of the line, the diagonal of the box being the original line. If BF is added the box is filled in the foreground colour.

The following program shows a triangle
subroutine in use. The co-ordinates of the three
points are chosen at random and the colours are
selected. To colour the triangle in, a point inside
in this case the points of the triangle are chosen
at random. So which point lies inside the triangle?
The point whose co-ordinates are the average
of the X values, $(XA + XB + XC)/3$ and the average
of the Y values, $(YA + YB + YC)/3$ will certainly lie
inside.

```

10 REM TRIANGLES MUPARDSON MAY 80
11 PMODE5:SCREEN16:PC15
12 FOR C=1 TO 4: COLOR C,1
13 XA=PMODE5(XH):YA=PMODE5(YC):XA0=XA
14 YL=PMODE5(YL):YL0=YL:YL1=YC:YL2=YH
15 GOSUB9:PAINTERA:XA=XA0:YA=YL0
16 PA=YA-YB:PC16,C:NEXT
17 GOTO9
18 UMODE,XA,YA=XA,YA,PA,PB,PBET
19 YA=(XA,YC):PA=LINE-(XA,YA),
20 PA=PA:PC16,C:NEXT

```

Writing a lot of LINE commands can be tedious. One way out is to write the command as a subroutine and change the values of the end points either by defining them directly in the program e.g. XA = 100, or, by reading them from DATA statements.

The following demonstration program shows how DATA statements can be used with a LINE subroutine to make a picture out of filled boxes. The COLOR command allows the boxes to be different colours. Only when a variable is changed does it's new value have to be READ. The program starts with 100 boxes and then RUN should reveal one with label "box".

Add these lines to the previous program:

29 PDF 1-1 TO 3D STEP2COLOR COLORLINE+LM=1-138
+LM1-1.PDF
30 CO-FINCH2000.DXF
31 PDF 1-1 TO 3D COLOR COLORLINE 09+LM=1-
138+LM1-1.PDF
32 CO-FINCH2000.DXF
33 CO-FINCH2000.DXF

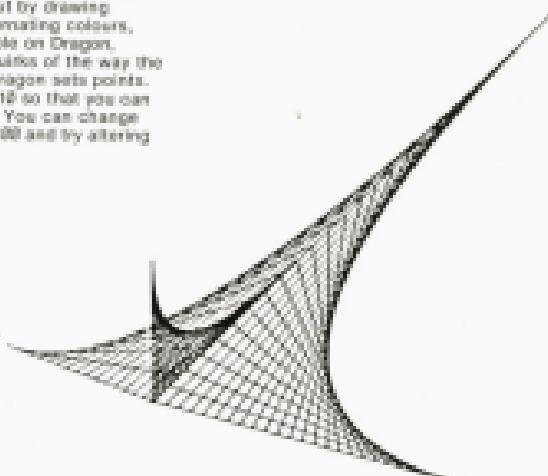
There is another sort of shading available with a LINE command. This involves joining points along two lines. Each line is divided into many equal intervals and the points are joined as in peg and sail designs. The effect is of a moulded shape. The following program shows this effect.

```

1 IBM CONCORDE A MUSYRES
10 PMOD04:1:00000000000000000000000000000000
20 FOR J=1 TO 3
30 READ R,A,B,C,D,A,B,C,C,D:FOR I=0 TO 5
40 Z=UNPK(X,I):C=APK2(Y,I)+D-BPK2(Z,A,A)
50 +BPK2(A,A,Y,I)+BPK2(B,B,Y,I)+BPK2(C,C,Y,I)
60 UNPK(X,I)=UNPK(SETNEXT,I)
70 GOTO30
75 DATA 10,10,10,100,100,100,100,100,100
80 DATA 10,10,10,100,100,100,100,100,100
85 DATA 10,10,10,100,100,100,100,100,100

```

Another way to change the co-ordinates is in a FOR...NEXT loop. This works well when the coordinates change in a regular way. The next few lines use this method to shade the top and sides of the 'box'. Five PAINT commands, together with PAINT could have been used but by drawing lines very close together in alternating colours, new colours, not readily available on Dragon, appear. These are created as quarks of the eye combines colour and the Dragon sets points. The colours are defined in line 10 so that you can experiment with different ones. You can change the step length in lines 70 and 100 and by altering the SCREEN.





YOUNG USER PAGES

NINE-NINE-NINE

Does anyone know what sound a dragon makes? Dogs bark, cows moo, and E.T. says 'goshen', but what do dragons do? The sort of dragons that live in lairs and harass villages by burning down houses whatever theyough, that is. Of course the sort that live in houses near the television only barkly the old folks (Over 30s). We know these Dragons sing, make crunch noises and bullet whizzes. But you haven't really got a well-trained Dragon until it makes noises that you want it to make. You can use the SOUND command to tell your Dragon to make noises. Type in SOUND 88, 32 followed by ENTER. You should hear a middle-C lasting about a second. If you don't then try turning up the sound or adjusting the volume on your TV!

Try using different numbers. The first determines the pitch and can be from 1 to 255. The second number controls the length of the note and must be greater than 0.

If you want to hear all possible pitches you can put the SOUND command into a loop. Doing so—

18 PQM P = 1 To 200
29 SONND P.1
30 NEXT
31 32

The loop takes *P* through all the values from 1 to 395 increasing one at a time. We can change the loop to take *P* through all values from 395 to 1 decreasing 1 at a time. Just change line 18. Coming down.

• 1995 EDITION •

Perhaps you want to play the same note over and over again. Too.

10 ROM J = 1 TD 6
20 SOUND 1974
30 1974

This loop takes *J* through values 1 to 5 but *J* isn't mentioned inside the loop so the program just does the same thing five times. How dull! Let's change line 20 to

ANSWER QUESTIONS

Run this. Does it remind you of anything? Now add these lines to give you the familiar drop-in switch.

48 SOUND TEST-ROUND 148
49 P08.1-1 TO 1
48 SOUND TEST-ROUND 148
50 NEXT

You can make very small changes in pitch using the SOUND command, but you need a very good ear to decide which number gives which note. However with the PLAY command you use letters CDEFGAB for a scale (together with sharps (#) and flats (-), or alternatively, numbers separated by semi-colons 1,2,3,4,...,11,12 to play successive semitones. This is the way to make your Dragon tuneful. There are lots of instructions available and page 113 of the Dragon Primer lists them all.

To make the siren sound using the PLAY command I decided to use B and B flat above middle C (B2) played very rapidly (T250) to give the siren effect. I played them 16 times to get the right length of note. For the lower note I used F and F# but you can change the notes to suit yourself. I wanted the siren to start quietly and come louder so the volume had to be increased. I could write lots if strings:

10 FOR L=1 TO 10
20 PRINT "V" = 0.73111
30 END

1.1.1. Logarithmic

When you use several loops in a program it's wise to state which variable you want when you end the loop with NEXT. Hence NEXT L.

How is the program for a given setting named?

```

10 FOR L=1 TO 18
20 FOR I=1 TO 15
30 PRINT "V"+STR(L)
40 PRINT TAB(20-L)+"-"
50 NEXT I
60 FOR I=1 TO 15
70 PRINT TAB(I)+"F"
80 NEXT I
90 NEXT L

```

If you want to carry on and make the siren go past 1000

100 FOR I=20 TO 1 STEP -1

and a drop in pitch say A and B flat with F and E. Now a challenge! Can your Dragon ring like a telephone? If it can, send your program to us and we will reward the best entry with free Dragon software! For details, see Young Users Competition.

Now follows a tiny program to develop your sounds by changing one line.

10 PI = "TOMASICA"
20 PLAY PI:GO TO 30

You must use the BREAK key to stop this program. Here are some other ideas for line 10:

10 PI = "TOM ABECA"
10 PI = "TOM ABEPT"
10 PI = "TOMO 1ABEPIP1"
10 PI = "TOMO 1ABEPIP1"
10 PI = "TOMOS ABEPIP1"
10 PI = "TOMSSCCCBFDPCOCBEPB" spaceship sound
10 PI = "TOMEDOCBCCCBEGFB" noise racing test
10 PI = "TOMCCCBCCCBEPB" never mind
10 PI = "TOMCCCBCCCBEGFB" hesitate mind
10 PI = "TOMGPIBT" spaceship No sound

Go PLAY AWAY!



WIN 4 SOFTWARE CASSETTES (COMPETITION)

If you are twelve years old or under and would like to win some Dragon software then devise a program on your Dragon 32 that uses the computer's SOUND or PLAY commands to make a telephone ringing. The program should not be more than twenty lines long.

The young programmer judged to have devised the neatest program will win 4 cassettes of their choice from the software listed on the back page and some posters illustrating some popular Dragon games. Send your entries to The Editor of the Newsletter at Dragon Data, 31st August 1983. The editorial decision is final.

The editor, The Newsletter, c/o Dragon Data Ltd, Kestrel Industrial Estate, Margam, Port Talbot, West Glamorgan.

DRAGON PUZZLE 2



This program plays a tune but first you must get the lines in the correct order. On the right are clues to the missing line numbers. Put the answers in the spaces left for them and type RUN. I'm sure you will recognise the tune.

1 DRAGON PUZZLE 2

...PLAY"100010001000"

... see the start of World War 1

...PLAY C1001 = 1102

... is quite a score

...PLAY"1400100000000000"

... is the year Columbus discovered the Bahamas

...PI = "14_0" PI = "14_1"

... is a baker's dozen

...PLAY"1800100000000000"

... is the year of the Dragon

...PLAY"1400100000000000"

... and World War 1 ended

...PLAY PI = 0:NEXT

... the year Harold saw no more

...PI = "1400000000000000"

... coming of age

SOLUTION TO DRAGON PUZZLE 1

10CLS:PRINT"DRAGON PUZZLE"

20 PRINT"10001000"

30 PRINT"100010001000"

40 PRINT"1000100000000000"

50 PRINT"1000100000000000"

60 PRINT"1000100000000000"

70 PRINT"1000100000000000"

80 PRINT"1000100000000000"

90 PRINT"1000100000000000"

100 FOR I=1 TO 3000:NEXT

110 FOR I=1 TO 16:P=PIE*PI/16*I

111 PRINT@IPI + 0.5PI + 0.5:

112 NEXT:PRINT@IPI,"

(See YOUNG USERS PAGE in previous issue)



RELEASES from the Dragon's Lair

Here are 24 new titles in the official Dragon Data Software range that should be appearing in the shops about now. Watch out for further Software news in STOP PRESS.

M30529 STALAG 4MO

No joysticks. Two adventure games. In stalag, your goal is to escape, alive, from a prisoner of war camp before it is bombed. In ero, you are searching for the money left by your late aunt. To prove that you deserve the fortune that has been left to you, your aunt has hidden the cash.

M30534 MANSION OF DOOM

No joysticks. Mansion of doom is an adventure game in which you have been chosen to receive the crown princess. Melding with the townspeople of her village in Transylvania, you must rescue the princess from the mysterious Count von Steinoff. His mansion, on the edge of the village, is replete with legend. The count himself has never been seen in daylight.

M30533 POSEIDON ADVENTURE

No joysticks. Poseidon Adventure is an adventure game in which you are aboard the luxury liner S.S. Poseidon. A huge underwater earthquake has caused a tidal wave which has capsized the ship. It is floating, bottom-up on the surface and taking on water. Your goal is simply to get out alive.

M30533 DREAMBUG

No joysticks. Dreambug is a monitor (debugger) and disassembler for the Dragon 32 which is designed to be used in conjunction with dream (editor/assembler) cassette for the Dragon 32.

M30533 FINAL COUNTDOWN

No joysticks. Final Countdown is an adventure game in which your mission is to prevent a nuclear missile from being launched and starting world war III. You begin the game outside a missile base which has been evacuated after a terrorist General has started the countdown on a missile.

M30530 TIMSCRIPT

No joysticks. Printer required. A speed writer which will enable you to use your Dragon to produce continuous text, such as business letters, quickly and easily. Timscript allows frequently used words and phrases to be replaced by two letter codes, considerably reducing the number of keystrokes required. The codes are automatically expanded when they are typed.

L30515 DREAM

No joysticks. A professional quality screen editor and assembler for use in the production of assembly language programs and subroutines.

E30501 ALLDREAM

EDITOR/ASSEMBLER

No joysticks. A screen editor, assembler, disassembler and monitor together on a cartridge. The package will enable the user to create and debug assembly language programs or assembly language subroutines to be called from within basic programs.

L30520 PIXEL EDITOR

No joysticks. This program will enable quick and easy creation and editing of graphics shapes. Each individual pixel can be accessed in order to produce detailed pictures and character sets.

L30521 HIDE AND SEEK

No joysticks. Hide and seek is a program designed by experts to aid short term memory and develop early reading skills. The program has a range of difficulty levels to suit children between the ages of four and eleven, and, as well as being educational, is a program which children will enjoy using.

L30518 NUMBER PUZZLER

No joysticks. Number puzzler is a program designed by experts to develop childrens powers of mental arithmetic. The program has a range of difficulty levels to suit children between the ages of four and eleven and, as well as being educational, is a program which children will enjoy.

L30517 NUMBER GRIPPER

Joysticks optional. As for number puzzler.

M30526 CIRCUS ADVENTURE

No joysticks. Circus adventure is an adventure game designed especially for children between the ages of 4 and 8. The program incorporates a number of user inputs to encourage keyboard familiarity and presents the child with a series of choices to be made to encourage decision making skills. The program is designed to be non-intrusive and children will enjoy using it.

M30529 SCHOOL MAZE

No joysticks. As for circus adventure.

M30510 DRAGON SELECTION 3

Joysticks required for aliens. Dragon selection 3 is a collection of four games programs: Money, Detective, Aliens, Hangram.

M30511 DRAGON SELECTION 4

No joysticks. Dragon selection 4 is a collection of three party games for children. Password, Lucky Dip, Composer.

J30111 RAIL RUMBLE

No joysticks. A fast moving video game, rail rumble is a race against time to cross a series of tracks, avoiding trains and hazards to rescue Herman Hobo.

M30519 EL DIABLO

No joysticks. El Diablo is an adult adventure game. You are wandering in the desert trying to regain your lost image powers before confronting El Diablo.

M30527 STORM ARROWS

Joysticks required. Storm arrows is a fast moving, multi-screen maze game. Your task is to defend yourself against hostile arrows while maintaining your energy supply.

K30112 GALAXY ATTACK

Joysticks required. As wave after wave of enemy ships attack, you must defend your ground base against them. The enemy craft fly in convoy formation but they will disband in order to attack.

K30532 SHARK TREASURE

Joysticks required. You have discovered the lost wreck of a ship which sank many years ago with a cargo of gold bars. The only thing between you and a fortune is a swarm of huge sharks. Shark treasure is graphic video game.

J30110 DOODLE BUG

Joysticks required. Doodle bug is a colourful, fast action maze game. An increasing number of enemies have to be avoided together with a number of dangerous obstacles.

K30535 SHUTTLEZAP

Joysticks required. Orbiting the earth your task is to grab as many satellites as you can, before landing safely back at base. This program has great graphics and what's more - it talks to you.

K30114 WHIRLYBIRD RUN

Joysticks required. Whirlybird run is a fast moving arcade type game with multiple screens, different types of attacker and a maze to be negotiated in the final stages.

DRAGON SOFTWARE



SOFTWARE AVAILABLE FOR THE DRAGON 32

M 30000 DRAGON SELECTION ONE

Four games for the younger user. Written in BASIC, they can be listed and edited.

M 30011 DRAGON SELECTION TWO

Collection of utilities. Create your own data base, write your own tunes...

M 30012 QUEST

Adventure game in a medieval setting. Defeat Moonlock, master of the dark castle.

M 30000 MADNESS AND THE MINDSAVER

A real-time adult adventure game.

M 30001 PERSONAL FINANCE

Keep track of family finances.

M 30002 GRAPHIC ANIMATOR

Create simple cartoons on the screen and animate them by flipping through the pages. Joysticks required.

M 30003 COMPUTER VOICE

Your Dragon will talk with this voice synthesiser.

M 30007 EXAMPLES FROM THE MANUAL

30 examples from the programming manual.

M 30008 CALIBUTO ISLAND

An adventure game. Return the hidden treasures to its rightful place.

M 30009 BLACK SANCTUM

An adventure game. Overcome the forces of black magic.

M 30012 TYPING TUTOR

Improve your speech and accuracy.

M 30013 DRAGON MOUNTAIN

An adventure game. Defeat the guardians of the treasure hidden in the mountain.

M 30014 ITAG

Race your opponent through a constantly changing maze to the final flag. Joysticks required.

M 30018 CL DRAGLORD

An adventure game set in the desert.

J 30100 IMPERIAL

A challenging shooting game, based on the popular arcade game, one or two players. A high resolution game in black and white. Joysticks required.

J 30101 METEOROID

Guide your ship through treacherous asteroid belt. A game requiring skill, fast reactions and concentration. A high resolution game in black and white. Joysticks optional.

J 30102 COSMIC INVADERS

Dragon version of the famous arcade game.

J 30103 GHOST ATTACK

Maze game for one player. Joysticks required.

J 30104 CAMEL HUNTER

Descent into the maze of caves in search of gold. Joysticks required.

J 30105 STARSHIP CHAMPION

Protect your planet from the attacks of the Goliathians. High quality arcade game with superb graphics and sound. Joysticks required.

J 30107 ASTROBLAST

Defend your ship against waves of attackers. A high resolution game in black and white. Joysticks required.

J 30108 CHESS

Nine levels of play, from beginner to master.

J 30111 RAIL RUNNER

Move Bill Switchman across the tracks, avoiding trains, to rescue Herman Hobo.